

**METHOD AND APPARATUS FOR PROVIDING DATA MANAGEMENT FOR A  
STORAGE SYSTEM COUPLED TO A NETWORK**

**Field of the Invention**

5        This invention relates generally to the field of information storage devices and more particularly to a method and apparatus for managing access to data in a storage device.

**Description of the Related Art**

Computer systems generally include one or more host processors and a storage system for  
10    storing data accessed by the host processor. The storage system may include one or more storage devices (e.g., disk drives) to service the storage needs of the host processor. Disk drives may include one or more disks of a recording media, such as a magnetic recording medium or an optical recording medium.

In a typical computer system configuration, a bus provides an interconnect between the  
15    host processor and the storage system. The bus operates according to a protocol, such as the Small Component System Interconnect (SCSI) protocol, which dictates a format of packets transferred between the host processor and the storage system. As data is needed by the host processor, requests and responses are forwarded to and from the storage system over the bus.

With the growth of networked computer systems, multiple hosts have been coupled over  
20    a network to a shared data storage system. Fibre Channel is an example of a network that can be used to form such a configuration. Fibre Channel is a network standard that allows multiple initiators to communicate with multiple targets over the network, where the initiator and target may be any device coupled to the network. Using a network, multiple hosts are able to share access to a single storage system. One problem with coupling multiple hosts to a shared storage  
25    system is the management of data access at the storage system. Because multiple hosts have access to a common storage system, each host may physically be able to access information that may be proprietary to the other host processors.

Various techniques have been implemented to manage access to data at the storage system. For example, certain portions or zones of memory at the storage system may be  
30    dedicated to one or more of the hosts. Each host is 'trusted' to access only those portions of memory for which it has privileges. However, such an approach is vulnerable to the individual actions of each of the hosts. As a result, such a data management method may not be sufficient to protect data from unprivileged accesses.

09107618-063098

### **Summary of the Invention**

According to one aspect of the invention, a data management method for managing access to a storage system by at least two devices coupled to the storage system includes a step of selectively servicing, at the storage system, a request from one of the at least two devices for  
5 access to a portion of data stored at the storage system responsive to configuration data indicating that the one of at least two devices is authorized to access the portion of data.

According to another aspect of the invention, a computer readable medium includes a first data structure to manage accesses by a plurality of devices to volumes of data at a storage system, the first data structure comprising a plurality of records corresponding to the plurality of  
10 devices, each record of the plurality of records corresponding to one of the plurality of devices and including configuration information identifying which of the volumes of the storage system the one of the plurality of devices is authorized to access.

According to another aspect of the invention, a storage system includes at least one storage device apportioned into a plurality of volumes, a configuration table to store  
15 configuration data identifying which of a plurality of devices coupled to the storage system are authorized to access each of the plurality of volumes, and a filter, responsive to the configuration data, to selectively forward to the at least one storage device requests for access to the plurality of volumes received from the plurality of devices.

20

### **Brief Description of the Drawings**

Figs. 1A, 1B and 1C illustrate exemplary network configurations in which the data management aspect of the present invention can be employed;

Fig. 2 illustrates one embodiment of a packet that can be distributed between devices coupled in the networks of Figs. 1A, 1B or 1C;

25 Fig. 3 is a block diagram illustrating exemplary components of a host processor and storage system which may be coupled in the networks of Figs. 1A, 1B or 1C, the storage system including an adapter having elements for filtering requests issued by the host processor according to one embodiment of the present invention;

Fig. 4 is a block diagram of one embodiment of a configuration data structure that may be  
30 used to store filtering information for use by the storage system of Fig. 3;

Fig. 5 is a block diagram of one embodiment of filtering data that may be used to filter requests at the storage system of Fig. 3;

Fig. 6 is a block diagram illustrating exemplary components of a host processor and storage system that may be coupled in the network of Figs. 1A, 1B or 1C, the storage system including logic for authenticating transactions by the host according to one embodiment of the present invention;

Fig. 7 is a flow diagram illustrating one method of distributing encryption keys between a host processor and storage system of Fig. 6, so that the encryption keys can be used to authenticate the identity of the host processor;

*Figs. 8a-c are*  
Fig. 8a is a flow diagram illustrating one method of authenticating the identity of a host processor when establishing a connection between the host processor and storage system of Fig.

6 in accordance with one embodiment of the invention;

~~Fig. 8b is a flow diagram illustrating one method of authenticating the identity of a host processor for each of a series of communications between the host processor and storage system of Fig. 6 according to one embodiment of the invention;~~

Fig. 9 is a diagram illustrating exemplary components that may be included in an authentication table of the storage system of Fig. 6 and that may be used to authenticate the identity of host processors according to one embodiment of the invention;

Fig. 10 is a flow diagram illustrating one method of performing a checksum of data transferred between an initiator and target in the networks of Fig. 1A, 1B and 1C for validating data transfers in one embodiment of the invention; and

Fig. 11 is a flow diagram illustrating one method of validating a data transfer between an initiator and target using the checksum of Fig. 10 according to one embodiment of the invention.

### Description of Detailed Embodiments

The present invention is directed to a data management method and apparatus for managing accesses by multiple devices (e.g., host processors, file servers and the like) to data at a shared resource (e.g., a shared storage system). According to one embodiment, the shared resource selectively services requests from the devices for portions of data at the shared resource in response to configuration data associated with each of the portions of data at the resource.

In one embodiment, data at the shared resource is apportioned into volumes.

Configuration data identifies which volumes of data are available for access by each of the devices coupled to the resource (e.g., over a network). The shared resource includes a filter that selectively forwards a request for servicing depending upon the identity of the device issuing the

request and the configuration data associated with the volume to which access is sought. The filter forwards only those requests for volumes that the device has privileges to access. Requests to volumes for which the device does not have privileges are not serviced.

Filtering requests at the resource allows the control of the data management to be centralized in one location, rather than distributed throughout the network. Centralizing the data management control at the storage system removes the need to trust the hosts seeking access to the storage system to only access certain portions of data.

In one embodiment of the invention, in addition to filtering requests to support data management at the resource, security protection may be added to further secure the data at the resource. Because filtering is performed in response to the identity of the device initiating the request, data security may be compromised if a device falsely represents its identity to gain access to the resource. In addition, data transfers between the device and the resource may be corrupted by other devices coupled (e.g., over a network) to the resource. In one embodiment, an authentication method and apparatus is provided to verify that the device that is represented as the device issuing the request is truly the device that issued the request. The authentication method may therefore be implemented to overcome security problems associated with a device mis-representing its identity to obtain data at the resource. In another embodiment, a validation method and apparatus is provided to ensure that information, transferred between the device and the shared resource is not corrupted (either inadvertently or intentionally) during transit.

A data management system for managing data at a resource may use any or all of these filtering, authentication and validation techniques. One exemplary system wherein the data management method and apparatus of the present invention may be employed is in a networked computer system, wherein the devices are host processors or file servers coupled to the network, and the shared resource is a storage system (e.g., a disk device storage system). It should be appreciated that the use of a network, host processor or shared disk device is not a limitation of the present invention, and that such a system configuration is described below solely for purposes of illustration.

In one embodiment, one or more hosts may be coupled to one or more storage systems using a network, with requests and responses being forwarded to and from the storage systems over the network according to the protocol of the network. In addition, each host and storage system may include one or more ports for interfacing the host or storage system to a corresponding one or more networks. As described above, in one embodiment, each storage

09107618-063098

system in the network includes logic for filtering received requests to manage data accesses to the storage system.

One exemplary network in which the present invention may be employed is a Fibre Channel network, although the present invention is not limited to the use of the Fibre Channel network or any other particular network configuration. Three example Fibre Channel network configurations are illustrated in Figs. 1A, 1B, and 1C. Fig. 1A illustrates a network 10 arranged in a loop configuration, where all devices in the network are coupled together in a single loop. In Fig. 1A, three host processors 12, 14 and 16 are shown coupled to a storage system 20 by a hub 18a. Internally, the hub is arranged in a loop configuration. Communication between the devices, over the busses 15a-15d, is performed by passing data packets from one device to the next in the loop. Fig. 1B illustrates a network 30 arranged in a fabric configuration, where all the devices are coupled together by a hub 18b. Internally, the hub is arranged as a switch. Communication between pairs of the devices 12, 14, 16 and 20 in the network 30 is controlled by the hub 18b. The data management method of the present invention may be employed in networks arranged in the loop or fabric configurations illustrated in Figs. 1A and 1B, or alternatively in other network or resource sharing configurations. For example, the data management aspect may be employed in the network illustrated in Fig. 1C. In Fig. 1C, a host/storage system configuration is shown where the storage system includes two ports, each of which interfaces the storage system to a different network. In Fig. 1C, a first port (Port 0) is coupled to a fabric network 30 and a second port (Port 1) is coupled to a loop network 10.

As mentioned above, the data management aspect of the present invention configures volumes of data at the storage system 20 according to the identity of the host devices coupled to the storage system. The configuration data that is used to manage the allocation of volumes to different hosts may be provided, for example, by a system administrator of the network. The system administrator tracks the host devices that are coupled to the network and the available volumes at the storage system. As a new host device enters the network, the system administrator allocates storage system volumes to the host. The number of volumes allocated to the host may be based on a requested number of volumes, or alternatively may be based on historical data requirements of the host. The system administrator may be implemented in software, executing on one of the devices or storage systems in the network, and may include a graphical user interface to enable users to monitor the availability and assignment of volumes to

09107618-063098

different hosts in the network. The present invention is not limited to any particular implementation of the system administration.

Generally, as each device enters the network it queries the network to identify the other devices coupled to the network. Each device that responds to the query returns one or more identifiers for the device. For example, the identifier may include the world wide name (WWN) assigned to the device by the manufacturer of the adapter board using a naming convention. The identifier may also include a source identifier (ID). Both are unique identifiers for the device, however the source identifier is generally a shorter identifier than the WWN. The source identifier may identify the device (e.g., a host processor) and the port of the device that is coupled to the network. Thus, if multiple ports of the particular device are coupled to the network, an identifier may be returned for each port. When the query operation is complete, each of the devices has knowledge of the other devices coupled in the network. Once each device has knowledge of the other devices in the network, this information can be used to exchange data between devices.

Requests and responses to and from the storage system 20 are forwarded over the networks 10 and 30 according to the network protocol. The Fibre Channel interconnect is capable of carrying multiple interface command sets. Accordingly, the devices that are coupled together using the Fibre Channel network may communicate using any of a number of higher level protocols including Internet Protocol (IP), Small Component System Interconnect (SCSI) protocol or any of a number of other protocols provided that the interfacing devices have knowledge of the type of protocol that is being used on the particular Fibre Channel interconnect. Certain types of devices have historically been designed to communicate using certain protocols. For example, host processor devices have historically communicated with storage systems using the SCSI protocol. Thus, devices coupled using the Fibre Channel network may communicate with each other using the same protocols that have historically been used. As a result, existing interfaces of the devices require little re-design to couple to the Fibre Channel network.

Tunneling techniques typically are used to convert packets of any type of protocol into packets that may be propagated on the Fibre Channel network. Using tunneling, one packet, formatted according to a first protocol, is enveloped in a second packet formatted according to the Fibre Channel network protocol. Thus, a SCSI packet may be enveloped in a Fibre Channel packet by a host processor or storage system for transmission on a Fibre Channel network. One

09107E18-06309B  
860290-B1920F60

example of a packet formatted according to a SCSI protocol and enveloped in a Fibre Channel packet is illustrated in Fig. 2.

In Fig. 2, a Fibre Channel packet 50 includes a header portion 55 and a payload portion 60. The header portion 55 includes a source ID field 52, a destination ID field 53 and a length field 54. The source ID field 52 identifies a device in the network that initiated the transmission of the packet 50. The destination ID field 53 identifies a target device for receiving the packet in the network. The length field 54 identifies a number of bytes in the packet. In addition, other fields defined in the Fibre Channel specification also may be included in the header, although these fields are omitted herein for clarity purposes.

The source ID field 52 and destination ID field 53 are used in the exemplary system of Fig. 1C to identify particular host processors and the storage system. When a host issues a request packet to the storage system, the source ID identifies the host and the destination ID identifies the storage system. In accordance with one embodiment of the invention, the storage system uses the source ID field 52 of the packet to index into configuration data identifying which of the volumes of data the respective host has privilege to access.

As mentioned above, the source ID field for a host accessing the storage system 20 can be used to identify the host that issued the request to index into configuration data for the host at the storage system. The configuration data identifies the portions of the storage system that are accessible by the host. Thus, the configuration data can be used by filter logic at the storage system to selectively service the host's request. Exemplary components of a host and storage system that may be used to implement the method <sup>and</sup> apparatus for filtering requests based on a host identifier will now be described with regard to Figs. 3 - 5.

As described above, in one embodiment of the invention, the storage system determines whether a received request should be serviced based on the configuration data. A copy of the configuration data may be stored in a memory at the storage system. Periodically, the configuration data may be updated by the system administrator as hosts enter and exit the network. The storage system also includes logic for communicating over the network and filtering logic, coupled to the memory that stores the configuration data, for determining whether a request received from the network should be serviced. The above-described data management system may be implemented in a variety of ways and the present invention is not limited to any particular implementation. However, for purposes of clarity, one embodiment of a host

processor and storage system capable of providing this data management functionality is illustrated in Fig. 3.

Fig. 3 illustrates a host processor 12 coupled to a storage system 20 using a network 21. The network 21 may be, for example, a Fibre Channel network arranged in any of the configurations illustrated in Figs. 1A - 1C. The host processor 12 may be a multi-processing unit, including one or more central processing units such as CPU 40 coupled by a local bus 43 to a memory 42. One or more host bus adapters (HBAs) 45 and 45a are coupled between the bus 43 and the network 21.

Each host bus adapter (HBA) 45 and 45a operates to connect the host processor 12 to the network. The HBAs 45 and 45a translate data received from the CPU 40 into the format dictated by the protocol of the network. In addition, the HBAs 45 and 45a translate data received from the network in packet format into data in a format usable by the CPU 40.

The host bus adapter may be implemented using a combination of hardware resident on the HBA and driver software stored in the HBA or in the memory 42. Alternatively, the host bus adapter may be implemented either entirely in hardware or software. In one embodiment, the HBA 45 includes a processor 41 coupled to a storage system 49. The processor 41 controls the flow and format of data into and out of the HBA 45. The memory 49 is used to provide temporary storage of data as it is transferred to and from the network. The HBA generates packets for transmission over the network 21, with each packet including a source ID field identifying the particular HBA. Because multiple HBAs may be included at each host, multiple source IDs may be associated with the same host.

The storage system 20 includes storage devices 38a - 38d which may include one or more disk devices. Access to the storage devices 38a-38d is controlled through the use of disk adapters 36a-36d which may be implemented using a programmed processor or custom hardware design. In the embodiment illustrated in FIG. 3, a disk adapter is provided for each storage device 38a-38d, although alternatively a disk adapter may be coupled to more than one storage device. In addition, disk adapters may include secondary connections to the storage devices 38a-38d of another disk adapter 36a-36d to permit recovery from failure of one disk adapter by shifting its functions to the second disk adapter.

The storage devices 38a-38d are apportioned into volume sets. One or more of the volume sets may be made available to one or more of the HBAs 45, 45a or the host processor 12. In one embodiment, references to the volumes in the storage devices 38a-38d by the HBAs are



performed using logical unit numbers (LUNs). There need not be a one-to-one correspondence between the logical unit numbers provided by hosts and the physical addresses of the disk devices.

5 A configuration database 32 (Fig. 3) stores information regarding which ones of the HBAs have access to which ones of the volumes. As discussed above, in one embodiment information in the configuration database is received from the system administrator and is periodically updated as the configuration of the network changes.

10 An example of the types of data that may be stored in the configuration database 32 include a history table 69. The history table is apportioned into one block for each of the ports of the storage system. Each block in the history table includes a list of those hosts that have queried the port as they entered the network. The identification information for each host may include the WWN name of the host, the source ID of the host, or other aliases of the host. This identification information may be used when the host logs into the storage system to match an identifier of the host with configuration data for the host.

15 The configuration database 32 may also include a header portion 70 for mapping the HBAs to the available ports at the storage system. A volume allocation portion 72 may be provided for allocating logical volumes of data at the storage system 20 to different HBAs. A mapping portion 74 is provided for mapping LUNs to physical addresses of the disks. In addition, a filter table 76 is provided for controlling which HBAs have access to which of the LUNs. The filter table 76 is generated using the volume allocation and mapping information and includes a record for each HBA coupled any of the ports of the storage system. An example implementation of the filter table 76 is provided in Fig. 4. Each record 76a - 76n includes the WWN associated with the HBA, a flag indicating whether the volumes allocated in this entry are shared, and a LUN map identifying which of the logical volumes the HBA may access. In one embodiment, the LUN map is in the form of a bitmask with one bit allocated to each LUN in the storage system. In one embodiment, a bit in the bitmask is set to indicate that the associated HBA indicated by the WWN has access to the corresponding LUN, although alternatively the bit may be cleared to indicate access. In addition, alternative embodiments wherein the available LUNs are indicated differently may also be used.

25 30 The storage system 20 (Fig. 3) also includes a filter and adapter unit 34. The filter and adapter unit 34 translates packets received from the network into data blocks and control for forwarding to the disk adapters 36a-36d. In addition, the filter and adapter unit 34 performs a

95107518-063098

filtering function to ensure that only those HBAs with privileges are able to access volumes. Thus, rather than trusting that the HBAs will only access those volumes which they have been assigned, the filter and adapter unit 34 controls accesses to the disks by filtering out non-privileged requests.

5 In one embodiment, the filter and adapter unit 34 includes a processor 80 coupled to a memory 83. The processor is used to control the transmission and translation of data between the storage system 20 and the network 21. The memory 83 is used to store a transient filter table 84. The transient filter table is apportioned into a number of tables, one for each port of the storage system. Each time that an HBA initiates a connection with the storage system 20 over  
10 one of its ports, filtering information is copied from the filter table 76 in the configuration database 32 to the appropriate entry in the transient filter table 84. The filtering information may include the source ID of the HBA logged into the port, a flag indicating whether the volumes associated with this entry are shared, and a LUN map for the HBA logged into the port, where the LUN map is copied from the filter table 76 in the configuration database.

15 In one embodiment, the configuration data in the transient filter table 84 is accessed for each request. The address of the request is provided in Bus/Target/LUN format, where the Bus portion indicates the Fibre Channel network address of the storage system 20, the Target portion indicates the storage system port address, and the LUN represents the volume address of the request. The address is compared with the entry in the transient filter table 84 which includes the  
20 LUN map associated with the HBA. If the bit in the LUN map associated with the addressed LUN indicates that the HBA has access to the LUN, the request is forwarded to the disk adapters 36a-36d for servicing. If not, the request is ignored.

The size of the transient filter table 84 is related to the number of ports provided at the storage system, the number of HBAs supported at each port and the number of LUNs in the  
25 storage system. An example configuration of the storage system 20 may include sixteen ports for accessing 4096 LUNs, with each port capable of supporting accesses by thirty two different HBAs. Thus, when the transient filter table 84 is large, if it was stored in a single large memory the access time for each I/O request may be long.

In one embodiment, to increase the response time performance of the storage system 20,  
30 the transient filter table 84 is arranged to allow for quick retrieval of the access information for each HBA. Referring now to Fig. 5, one illustrative configuration of the transient filter table is shown. The transient filter table 84 is shown to include an array of records, such as record 400.

09107618-063098

One column of records is provided for each LUN in the storage system numbered in Fig. 5 as LUN0 to LUNx, where x+1 is the number of LUNs in the storage system. One row of records is provided for each port at the storage system. Each record includes a bitmap 402. The bitmap includes a number of bits corresponding to the maximum number of devices (HBAs) that can access each port. In Fig. 5, these bits are indicated as D1, D2 . . . Dn, where n is the maximum number of devices that may be coupled to any port.

During operation, as an I/O request is received at the storage system 20, the address of the I/O request is compared with the data in the transient filter table 84. The address includes an identifier of the HBA which initiated the request, and an address to the storage system portion that the host wants to access. This address includes, generally, a Bus/Target/LUN combination of fields. The bus field is the base address of the storage system 20 on the network, the target is the source ID of the port of the storage system to which the request is directed, while the LUN indicates the logical unit addressed by the request. The target information (row) and LUN information (column) are used to index into the transient filter table to obtain one of the records. The source ID for the requesting device (e.g., HBA 45 in Fig. 3) is then used to select one of the bits in the bitmap 402 of the record 400 selected by the Bus/Target/LUN address to identify whether or not the bit is set in the mapping. If the bit is set in the mapping, then the request is forwarded to the disks for servicing. If not, the request is dropped.

The transient filter table 84 may be stored in memory, as illustrated in Fig. 3, or alternatively implemented in hardware. While the configuration of the transient database described above provides one method of accessing HBA and LUN mapping information, alternative configurations may also be used. The present invention is not limited to this implementation. Rather, any configuration database arrangement that allows mapping data to be obtained from the database using a source identifier and a resource address may be used.

Accordingly, a data management system that reserves volumes of data for host processors based on host identifiers has been described. In one embodiment, the data management system is augmented to add security protection by authenticating each request to verify that it was truly sent by the host indicated by the host identifier received at the storage system 20. Authentication of the connections to the storage system 20 secures the data volumes allocated to each of the host processors by verifying that the host requesting a data volume is truly the host indicated in the source ID of the request packet. Simply relying on the source identifier of the host processor may not be a sufficient measure of protection. In some insecure environments, a processor may

issue a false source identifier in an attempt to access privileged data of another host.

Authentication thus verifies that the host processor is who it claims to be, and therefore should be allowed access to the data.

In one embodiment, to support the authentication of host requests, the storage system  
5 issues a number of unique, expected identifiers to each HBA of each host device prior to the  
initiation of transactions by the host device. The storage system maintains a copy of the  
expected identifiers. When the HBA associated with the host device initiates a series of  
transactions, it forwards the expected identifiers that were received from the storage system back  
10 to the storage system. The storage system compares each received identifier against the expected  
identifier. Because the storage system only forwarded the identifiers to the particular HBA  
associated with the host, if the correct identifier is received the storage system can verify that, in  
fact, the request was issued from the indicated host. In one embodiment, the level of security  
protection is further enhanced by using a random number for each identifier. In another  
embodiment, an encryption key, known only to the host and the storage system, is used to  
15 encrypt the identifier before it is sent to the storage system. The storage system may then  
decrypt the identifier using the encryption key. Encrypting the identifier prevents other devices  
that are coupled to the network from obtaining the identifier and gaining access to privileged  
data. Thus, a two-tiered level of authentication is provided, since each transaction has a unique  
identifier, known only to the issuing host, and the identifier is encrypted using a unique access  
20 key known only to the HBA at that host.

In one embodiment, the identifier information is transferred between the devices on the  
network (e.g., the host and the storage system 20) using selected fields of the packet. Referring  
again to the example packet of Fig. 2, illustrated in the payload portion 50 of the packet 50 are  
some of the fields that are used to communicate according to the SCSI protocol. These fields  
25 include a source session ID 62, a destination session ID 64 and a data field 65. According to the  
SCSI protocol, the source session ID and destination session ID are used to track an order of  
transactions between an initiating device and a target. The SCSI protocol allows a number of  
transactions to be simultaneously outstanding between an initiating device and a target device.  
For example, a host initiator may issue a number of I/O requests to different volumes in a target  
30 storage system. Depending upon the accessibility of each of the volumes, the order in which the  
responses to the requests are returned to the host may differ from the order of the issued requests.  
The session IDs are used to align requests with responses at the host. The session IDs provided

for each transaction may be arbitrary numbers as long as they are unique across a given time period. Because the session ID may be any arbitrary number, in one embodiment the session ID fields are used to transfer unique identifier information for each transaction.

The above-described authentication method may be implemented in a variety of ways and the present invention is not limited to any particular implementation. However, for purposes of illustration, an embodiment of a host processor 312 and storage system 320 including components for providing the above-described authentication functionality is shown in Fig. 6. In Fig. 6, the host bus adapter 345 stores in its memory 349 a data structure 60. The data structure 60 includes, for each storage system 320 coupled to the HBA, a public key 61, a private key 62, an access key 63, and a list 64 of one or more random numbers. Each of the keys is a field of bits which may be used to encrypt and decrypt data using techniques known to those of skill in the art. Encryption and decryption are performed by applying the bit values of the keys to bit values of data using logical operations associated with a selected encryption algorithm. The access key is a key that is dedicated to encrypting and decrypting authentication data transferred between the HBA and the associated storage system. In one embodiment, an initial value for the access key 64 is obtained using the public and private keys 62 and 63, respectively, as will be described in more detail below.

To support authentication, the storage system 320 includes a filter and adapter unit 334 that may include each of the features described with regard to Fig. 3, as well as a number of other components to support authentication. The filter and adapter unit 334 includes a random number generator 82 which is used to provide random numbers used in the authentication process for each of the HBAs coupled to the storage system. Coupled to the random number generator 82 is an encrypter/decrypter 85 and a comparator 87. The transient filter table 84, authentication table 86, random number generator 82, encrypter/decrypter 85 and comparator 87 are used in conjunction to authenticate connections to the storage system 20 using a method that will be described in more detail below.

As described above, the authentication method involves the transfer of identifiers between the host 312 and the storage system 320 over the network 21. In one embodiment, the identifiers are encrypted using an access key stored at the host 312 and at the storage system 320, although this is not a requirement of the invention. Fig. 7 illustrates one method of distributing an access key to the host 312 and the storage system 320.

Access keys may be provided in a number of ways, including allocating access keys to the storage system and host processors in advance of their entry to the network. Alternatively, an access key may be obtained after the host processor has entered the network in a manner described with regard to the data flow diagram of Fig. 7. In Fig. 7, operations performed by the HBA 345 (Fig. 6) are illustrated on the left hand side of the figure, operations performed by the storage system 320 are illustrated on the right hand side of the figure, and data flows are illustrated between the two sides. At step 100, the HBA 345 (Fig. 6) generates a public and private key pair. This can be done, for example, using known encryption software executing on the CPU 40. The public and private key pair is dedicated to the HBA and is not accessible by any other device coupled to the network 21 (Fig. 6). Data which is encrypted by the public key 61 can only be decrypted using the private key 62, while data that is encrypted by the private key 62 can only be decrypted using the public key 61.

At step 102, the HBA 345 forwards the public key 61 over to the storage system 320. In step 101, the storage system 320 loops, waiting for the key. When the storage system 320 receives the public key at step 101, it proceeds to step 103. In step 103, using the public key 61, the storage system 320 encrypts an access key for the HBA 345, forwards this encrypted access key 67 back to the HBA 345 over the network 21 (Fig. 6) and terminates. Because the HBA 345 is the only device in the network that has the private key 62 that must be used to decrypt any message encrypted with the public key, other devices that are monitoring the transactions on the network will not be able to decipher the encrypted access key 67 forwarded by the storage system 320. When it is determined at step 105 that the HBA 345 has received the encrypted access key 67, the process running on the HBA 345 proceeds to step 106, wherein the HBA 345 decodes the encrypted access key using the private key 62. In addition, at step 106, the HBA 345 stores the access key in data structure 60 with the public key 61 and private key 62 for later communications with the storage system 320.

Once the host processor has received the access key, it may begin the process of issuing I/O requests to the storage system 320. The processes by which a host processor issues an I/O request, and by which an I/O request is serviced by the storage system 320 in accordance with one embodiment of the invention will now be described with regard to the flow diagrams of Fig. 8a-b and Fig. 9.

In Fig. 8a, a flow diagram is provided for illustrating one embodiment of a method that may be performed to establish a connection between the HBA 345 and the storage system 320.

o In Fig. 8a, operations performed by the HBA 345 are indicated on the <sup>left</sup> ~~right~~ hand side of the figure, operations performed by the storage system 320 are indicated on the <sup>right</sup> ~~left~~ hand side of the figure, and data flows are shown in between. At step 110, the HBA 345 arbitrates for access to the storage system 320. At step 120, when the HBA 345 has won arbitration, it issues a control data block 112 to the storage system 320. The control data block 112 includes the source ID of the HBA 345 and indicates that the HBA requests to be serviced by the storage system 320. When it is determined at step 121 at the storage system 320 that there is an outstanding request from the HBA, the method proceeds to step 122 wherein in response to the host's request, the storage system 320 forwards a unique identifier 114 to the HBA 345. In one embodiment, the identifier is a random number provided in the destination session ID 64 of the payload portion 60 of the packet 50 as illustrated in Fig. 2. At step 122a, after the storage system has forwarded the random number to the HBA, the storage system 320 encrypts the random number 114 using the access key and stores the encrypted random number for later comparison.

When it is determined at step 123 that the HBA has received the random number from the storage system 320, at step 124 the HBA encrypts the random number 114 using the access key 63 (obtained using the process of Fig. 7), and returns the random number to the storage system 320 as an encrypted number 125. When it is determined at step 126 that the storage system 320 has received a response, at step 127 the storage system 320 compares the received random number 125 against the previously encrypted and locally stored version of the random number generated at step 122. The compare status 128 is forwarded to the HBA. If there is not a match, this indicates to the storage system that the host does not have the correct access key, and/or that the host does not know the correct random number and the connection between the host processor and the storage system is not authenticated. The HBA is not permitted to access data at the storage system 320, and the transaction is complete. If there is a match, then the host has the correct access key, and the correct random number and connection between the HBA and the storage system is authenticated.

Once the connection between the HBA and the storage system has been authenticated, the HBA may send an I/O request to the storage system 320 for servicing. In one embodiment of the invention, every I/O transaction is authenticated using a different identifier (e.g. random number). Because the SCSI protocol allows for multiple I/O requests from an HBA to be outstanding at any given time, the HBA keeps track of the outstanding I/O requests using a technique referred to as command tag queuing. Each I/O request is assigned a unique tag, using

the source session ID field 62 (Fig. 2). When a response is returned from the storage system, the tag is included in the session ID field of the response so that the response data may be aligned with the request using the tag.

The maximum number of I/O requests that an HBA may have outstanding at any instant in time is determined according to the command tag queuing protocol. Because each I/O request is assigned a unique identifier, each HBA should always include a sufficient number of identifiers to ensure that an identifier can be provided for each outstanding request. Therefore, according to one embodiment, as each device logs into the storage system a list of unique identifiers is forwarded to the device. The number of identifiers in the list corresponds to the maximum number of I/O requests capable of being outstanding at the device, although the present invention is not limited to this number. As each device issues a request to the storage system, one of the identifiers is used from the list. Each time that the storage system responds to a request, it returns a new identifier which is stored at the bottom of the list. Identifiers are selected in first in first out order from the list. As a result, a constant number of identifiers is maintained at the device to satisfy command tag queuing protocols.

One embodiment of a method of providing the identifiers to the host is described below. Because a unique identifier or random number will be associated with each I/O request, at the start of an I/O sequence, the HBA requests a number of random numbers equal to the maximum number of permissible outstanding I/O requests. The list of random numbers is stored as list 64 in data structure 60 (Fig. 6) of the HBA 345 and is additionally stored in the authentication table 86 of the filter and adapter unit 334 (Fig. 6) of the storage system 320.

Referring now briefly to Fig. 9, an example of elements that may be stored in the authentication table 86 (Fig. 6) is shown. For each of the HBAs that is coupled to the storage system 320, a data structure 89 is provided. The data structure may include a public key 89a, which is a copy of the public key 61 that was provided by the HBA to obtain the initial access key 63. In addition, the data structure 89 may include an access key 89b, which is a copy of the access key 63 stored at HBA 345. In one embodiment, each of the data structures further includes a list of random numbers. The random number in each entry of the data structure 89c corresponds to a random number that will be used to authenticate an associated transaction. How the random number is used to authenticate a transaction is described in more detail with regard to Fig. 8b.



In Fig. 8b, at step 184, the HBA issues a control data block 185 to the storage system 320. The control data block 185 includes control for notifying the storage system 320 that the HBA is going to initiate an I/O transaction, and that the connection has been authenticated. When it is determined at step 186 that a request has been received, the storage system 320  
5 decodes the request and at step 187 forwards a packet 188 including a list of random numbers to the HBA 345. At step 187a, the storage system 320 also encrypts the first random number using the access key and stores the encrypted number locally for later comparison with the encrypted number received in a request from the HBA.

When it is determined at step 189 that the HBA has received the packet 188, at step 190  
10 the HBA 345 checks to determine whether there is an outstanding I/O request. If so, at step 192, the HBA 345 encrypts the first random number using the access key, stored in field 63 of the data structure 60. The HBA 345 then inserts the encrypted random number in the source session ID field of the I/O request packet, and forwards the I/O request packet 194 to the storage system 320.

15 When it is determined at step 195 at the storage system 320 that an I/O request packet has been received, at step 196 the storage system 320 compares the encrypted random number received in the source session ID field of the I/O request packet 194 against the encrypted random number generated at step 187. Alternatively, the storage system 20 may simply decrypt the received random number and compare it against the expected random number. If there is a  
20 match, the storage system 320 services the I/O request, and forwards a response packet 197 back to the HBA 345. The response packet includes any data requested by the HBA, and a new random number which will be used once the numbers in the list 188 are exhausted. In one embodiment, the new random number is provided in the destination session ID field, while the previous, encrypted random number is returned in the source ID field for tracking purposes,  
25 although the use of specific fields is not a limitation of the present invention.

When it has been determined at step 198 that a response has been received from the storage system 320, the HBA retrieves the non-encrypted random number from the session ID fields of the packet 197 and stores the random number in the data structure 60. The process then proceeds to step 190, where the HBA initiates the next I/O request, using the next random  
30 number from the random number list 64 encrypted using the access key.

Thus, a method is provided wherein unique identifiers are used to authenticate a host at a storage system. Although the method has been described using random numbers, it should be

understood that this is not a requirement of the invention, and that any sequence of numbers that does not include a readily discernable pattern may alternatively be used. As discussed above, more than one HBA may be simultaneously connected to the storage system 320. Thus, as identifiers are distributed to coupled HBAs, their sequence will similarly be distributed among the variety of HBAs that are coupled to the storage system 320, thereby making the number distribution appear even more random and accordingly more secure. In addition, although encryption techniques have been described, it is not a requirement of the invention that identifiers be encrypted prior to transfer. Alternatively, an authentication technique may be used which does not include identifiers for each request but encrypts the entire request using the access key. It is also envisioned that a different access key may be provided for each transaction. The present invention may incorporate any combination of these authentication techniques and is therefore not limited to the specific embodiments discussed above.

The above techniques may be used to verify that a connection between an initiator and a target is authorized. In another embodiment, techniques are also used to ensure that the data that is received at the target is the data that was sent by the initiator. Accordingly, further security may be obtained by applying validation methods to ensure that commands and data are not corrupted during a transfer from the initiator to the target. The validation method may be provided to preclude other devices that are coupled to the network from interfering with transactions between the initiator and the target.

According to another aspect of the invention, a method for validating a data transfer between a source and a target over the network includes the step of maintaining, at both the target and the initiator, a digital signature of the transfer. The digital signature is in the form of a checksum. Each byte of data that is transferred between a target and an initiator is combined with the previous bytes of the transfer using a hashing function to form the checksum. The hashing function may be, for example, an exclusive OR function, or some derivative thereof. The data included in the signature may include only that data forwarded by the initiator to the target, only that data forwarded from the target to the initiator, or a combination thereof.

Referring now to Fig. 10, a flow diagram of one embodiment of a method for validating a data transfer between an initiator and a target is shown. The flow diagram is similar to that described with regard to Fig. 8b, but includes steps for validating the data transfer. At step 200, it is determined at the initiator whether or not there are any I/O requests. If so, at step 202, the initiator encrypts the random number associated with the transaction and forwards the I/O

request<sup>203</sup> with the encrypted random number in the source session ID, to the target. When it is determined at step 204 that the target received the request, at step 205 the target compares the received encrypted number against a stored encrypted expected random to determine a match. If there is no match, the target does not respond. If there is a match, at step 206 the target forwards a new encrypted random number and response data to the target. Also, at step 206, the target hashes the response data into the existing checksum to provide a new checksum to build the digital signature of the data transfer. The response 207 is received at the initiator in step 208. At step 209, the initiator also hashes the response data into the existing checksum to provide a new checksum to build the digital signature of the data transfer. The process of issuing I/O requests and generating checksums continues for the number of requests in the initiator/target connection. When it is determined at step 200 that there are no more requests, the process proceeds to the validation phase in step 210, illustrated in Fig. 11.

At step 210, the initiator encrypts the checksum using the access key, and forwards the encrypted checksum 212 to the target. The target receives the encrypted checksum, at step 214. At step 216, the target decrypts the checksum and at step 218 the target compares the decrypted checksum with the checksum generated by the target. The target sends a status packet 220 to the initiator indicating whether or not the checksums matched. If it is determined at step 222 at the transaction was valid (i.e, no data was corrupted), then the connection between the initiator and the target is completed. If the status 220 indicates that the transaction was not valid and that data was corrupted, then the initiator re-establishes the connection with the target and repeats the data transaction.

Thus, a data management method and apparatus has been described that provides three levels of data management. Filtering is provided to ensure that each host only accesses volumes of data for which it has privileges. Security may be incorporated using authentication methods to verify that each request was truly issued by the host indicated in the identifier of the request. Finally, validation may be provided to ensure that data is not corrupted during transit. A system using the data management method and apparatus of the present invention may incorporate one or more of these levels of data management independently, and thus the present invention is not limited to a system including all the recited data management techniques described above. Such a data management system provides increased control over data accesses by users at a host processor by limiting the data capable of being accessed by the host processor. Thus, while users at a host processor may be capable of accessing all of the data allocated to the host processor

09107618.063098

(depending upon the privileges associated with the user), neither the users or even the system administrator with the highest privilege at the host processor is capable of accessing data that is not allocated to the host by the storage system, regardless of the privileges of the user on the host.

5           Although the above discussion relates to an exemplary system wherein a host gains access to data at a storage system, the above described methods may be applied to any system wherein a resource is shared by multiple devices. Such systems include, but are not limited to any type of storage system including disk devices, tape devices, file servers and the like.

10           Having described several embodiments of the invention in detail, various modifications and improvements will readily occur to those skilled in the art. Such modifications and improvements are intended to be within the spirit and scope of the invention. Accordingly, the foregoing description is by way of example only, and is not intended as limiting. The invention is limited only as defined by the following claims and the equivalents thereto.

09107618.063098  
860E9D" BT920T60